Title of document

Hugo Ponthieu

Table of Contents

Abstract

This is the abstract of the document.

1. Global Architecture

1.1. Presentation

1.2. Schema

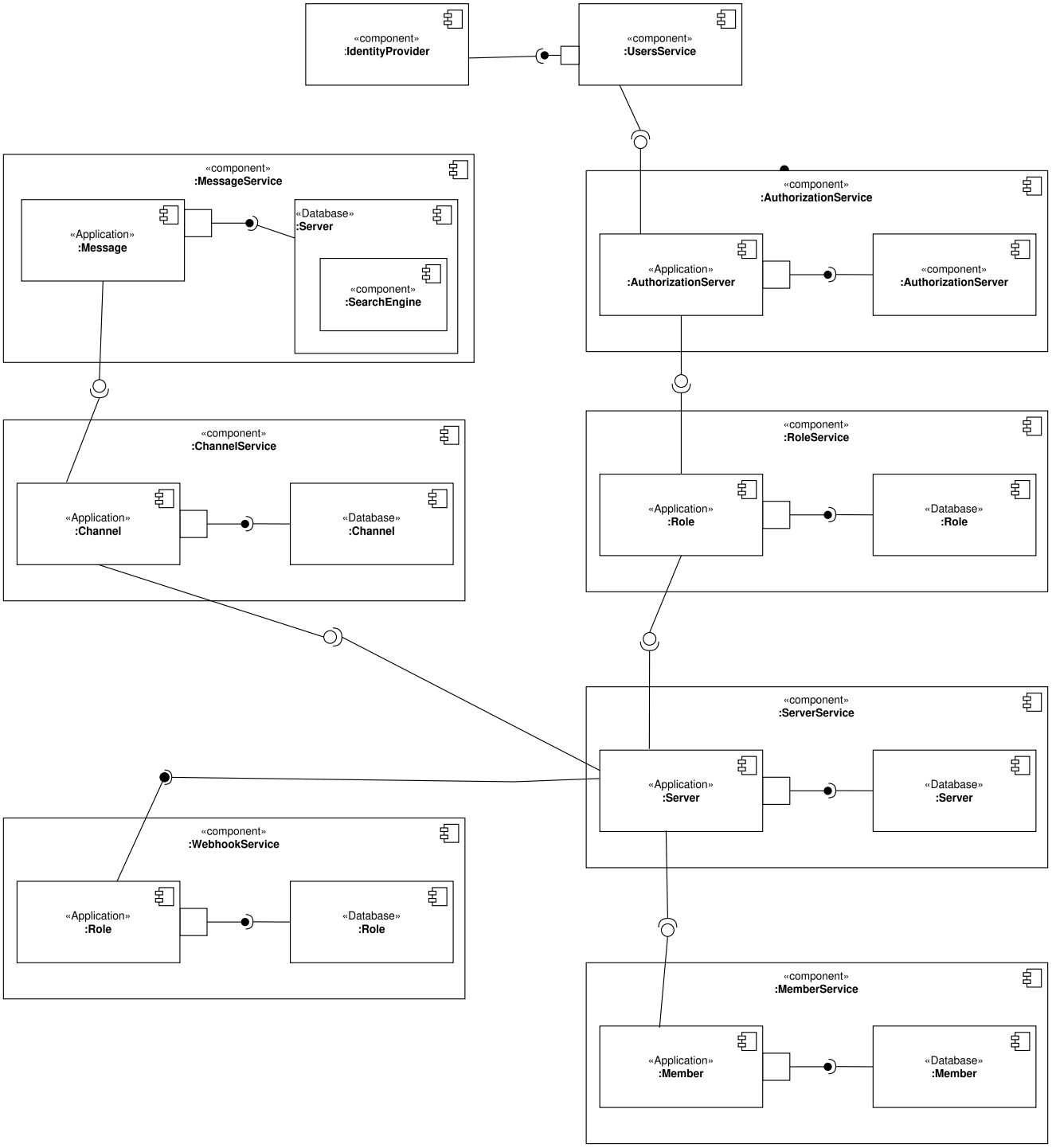


Figure 1. Overview of the application

1.3. Security

1.3.1. Authentication

Frontend integration of keycloak

Backend integration of keycloak

1.4. Introduction

Authentication is a critical aspect of any microservices architecture. In this document, we will discuss how to integrate Keycloak for authentication to enforce authentication policies at the gateway level.

1.5. Keycloak Overview

Keycloak is an open-source identity and access management solution. It provides features such as single sign-on (SSO), user federation, and social login. Keycloak is a suitable choice for our application due to its robust authentication capabilities and ease of integration with microservices.

As the user should be able to authenticate with their email and password, with their google account and their Polytech account from an LDAP Keycloak is suited for this task.

1.5.1. OAuth2 Overview

OAuth2 is an authorization framework that enables applications to obtain limited access to user accounts on an HTTP service. It works by delegating user authentication to the service that hosts the user account and authorizing third-party applications to access the user account. This is done without exposing the user's credentials to the application.

OAuth2 is suitable for microservice applications because it provides a secure and standardized way to handle authentication and authorization across multiple services. By using OAuth2, microservices can delegate the responsibility of user authentication to a centralized identity provider, such as Keycloak, and focus on their core functionalities. This approach simplifies the management of user identities and access control in a distributed system.

1.5.2. In our architecture

For example if a user wants to access a resource on a service, the service will redirect the user to the authorization server (Keycloak) to authenticate the user. Once the user is authenticated, the server will issue an access token to the user, which can be used to access the resource. This token is short-lived and can be revoked at any time, providing an additional layer of security.

From the access token the user will be able to access the service. To enforce the check of the access token the service will use the introspection endpoint of the authorization server.

We have to note that all service will have an upstream gateway that will check the access token of the user before forwarding the request to the service. This will ensure that only authenticated users can access the services.

Although the user will maybe need to be known by the service, in order to perform some actions. For example, getting the the list of its friends or direct messages. In that case the service will access directly the authorization server to get the user information.

1.5.3. Authorization

2. Implementation

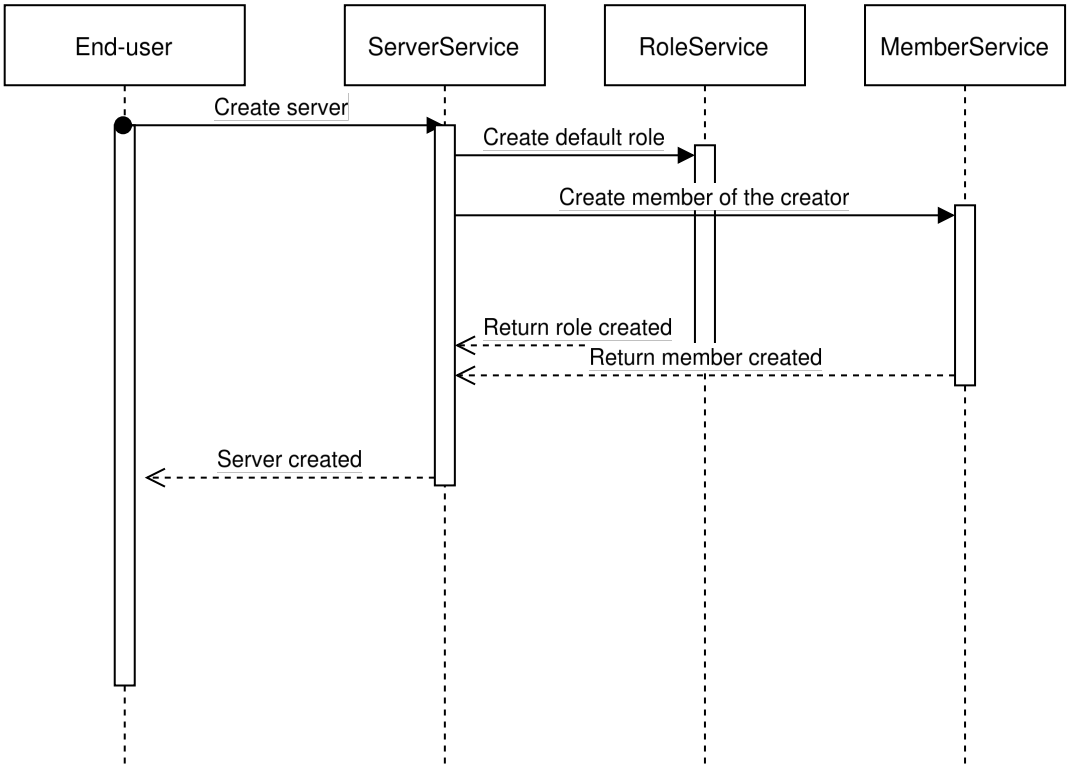
2.1. Deployment

2.1.1. Mesh Service

2.2. Protocols

Poc grpc with rust

2.2.1. Inter-service communication



2.2.2. Client communication

2.3. Services

2.3.1. Messages

Search

2.3.2. Users

2.3.3. Members

2.3.4. Roles

2.3.5. Authorization

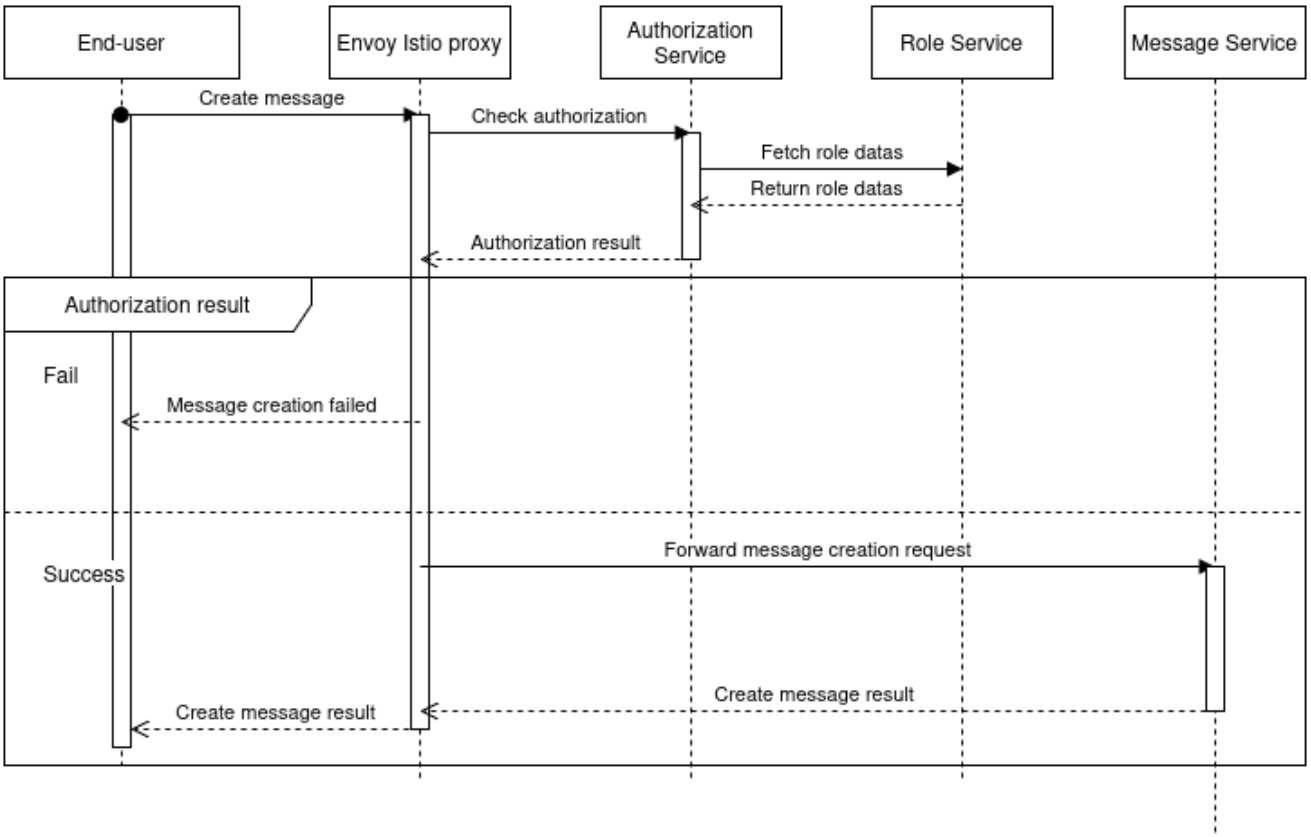


Figure 2. Create a message in a channel of a server and check the authorization

2.3.6. Servers

2.3.7. Channels

2.3.8. Messages

2.3.9. Webhooks

2.4. Backuping

2.5. Monitoring